IBM Compiler Optimization Arguments

May 30, 2003

Michael Stewart
NERSC User Services Group
pmstewart@lbl.gov
510-486-6648

Introduction

Why be concerned with optimization arguments?

What are the most useful optimization arguments?

Examples of the effects of various optimization arguments on benchmark codes.

IBM Default: No Optimization!

Can have very bad consequences:

```
do i=1,bignum
   x=x+a(i)
enddo
```

bignum stores of x are done with the default, no optimization.

When optimized, **store motion** is done: intermediate values of x are kept in registers and the actual store is done only once, outside the loop.

NERSC/IBM Recommendation

For all compiles - Fortran, C, C++:

-O3 -qstrict -qarch=pwr3 -qtune=pwr3

Compromise between minimizing compile time and maximizing compiler optimization.

With these options, optimization only done within a procedure (e.g. subroutine, function).

Numerical results bitwise identical to those produced by unoptimized compiles.

Drawback: does not optimize complex or even very simple nested loops well.

Numeric Arguments: -O2 and –O3

-O0 and **-O1** not currently supported.

-O2: Intermediate level producing numeric results equal to those produced by an unoptimized compile.

-O3:

- More memory and time intensive optimizations.
- Can change the semantics of a program to optimize it so numeric results will not always be equal to those produced by an unoptimized compile unless **-qstrict** is specified.
- No POWER3 specific optimizations.
- Not very good at loop oriented optimizations.

Most benchmarks achieve 90% or better of their maximum possible performance at the **–O3** level.

Numeric Arguments: -O4

Equivalent to "**–O3 –qarch=auto –qtune=auto -qcache=auto -qipa -qhot**".

Inlining, loop oriented optimizations, and additional time and memory intensive optimizations.

Option should be specified at link time as well as compile time.

If you are experimenting try "**-O4 –qnohot**" as well as "**-O4**".

Numeric Arguments: -O5

Equivalent to “**–O4 –qipa=level=2**”.

Full interprocedural optimization in addition to **–O4** optimizations.

Option should be specified at link time as well as compile time.

If you are experimenting, try “**-O5 -qnohot**” as well as “**–O5**”.

-qstrict: Strict Equality of Numeric Results

Semantics of a program are not altered regardless of the level of optimization, so numeric results are identical to those produced by unoptimized code.

Inhibits optimization (in principle) - does not allow changes in the order of evaluation of expressions and prevents other significant types of optimizations.

In practice, this option rarely makes a difference at the **–O3** level and can even improve performance.

No equivalent on the Crays.

-qhot: Loop Specific Optimizations

Now works with C/C++ as well as Fortran.

Loop specific optimizations: padding to minimize cache misses, "vectorizing" functions like sqrt, loop unrolling, etc.

Works best on loop dominated routines, if the compiler has some information about loop bounds and array dimensions.

-qreport=hotlist produces a (somewhat cryptic) listing file of the loop transformations done when **-qhot** is used.

Can double or triple compile time and may even slow code down.

Included by default with **–O4** or **–O5** compiles.

-qipa: Interprocedural Analysis

Examines opportunities for optimization across procedural boundaries even if the procedures are in different source files.

Inlining - Replaces a procedure call with the procedure itself to eliminate call overhead.

Aliasing - Identifying different variables that refer to the same memory location to eliminate redundant loads and stores when a program's context changes.

Can significantly increase compile time.

Many suboptions (see man page).

3 ipa numeric levels: **-qipa=level=n**.

-qipa=level Optimizations

Determines the amount of interprocedural analysis done.

The higher the number the more analysis and optimization done.

-qipa=level=0: Minimal interprocedural analysis and optimization.

-qipa=level=1 or **-qipa**: Inlining and limited alias analysis. **(-O4)**

-qipa=level=2: Full interprocedural data flow and alias analysis. **(-O5)**

-qarch: Processor Specific Instructions

-qarch=pwr3: Produces code with machine instructions specific to the POWER3 processor that can improve performance on it.

Codes compiled with **-qarch=pwr3** may not run on other types of POWER or POWERPC processors.

The default at the **–O2** and **–O3** levels is **-qarch=com** which produces code that will run on any POWER or POWERPC processor.

Default for **–O4** and **–O5** is **–qarch=auto**(=pwr3) on seaborg.

When porting codes from other IBM systems, make sure that the **–qarch** option is valid on seaborg.

-qtune: Processor Specific Tuning

-qtune=pwr3: Produces code tuned for the best possible performance on a POWER3 processor.

Does instruction selection, scheduling and pipelining to take advantage of the processor architecture and cache sizes.

Codes compiled with **-qtune=pwr3** will run on other POWER and POWERPC processors, but their performance might be much worse than it would be without this option specified.

Default is for no specific processor tuning at the **–O2** and **–O3** levels, and for tuning for the processor on which it is compiled at the **–O4** and **–O5** levels.

ESSL Library

Single most effective optimization: replace source code with calls to the highly optimized Engineering and Scientific Subroutine Library (ESSL) .

The ESSL library is specifically tuned for the POWER3 architecture and has many more optimizations than those that can be obtained with **–qarch=pwr3** and **–qtune=pwr3**.

Contains a wide variety of linear algebra, Fourier, and other numeric routines.

Supports both 32 and 64 bit executables.

Not loaded by default, must specify the **–lessl** loader flag to use.

-lesslsmp: Multithreaded ESSL Library

When specified at link time ensures that the multi-threaded versions of the essl library routines will be used.

No change required to source code.

Can give significant speedups if not all processors of a node are busy.

Default is to use 16 threads. Change the number of threads by setting the **OMP_NUM_THREADS** environment variable to the desired number of threads.

-qessl: Optimize Fortran Intrinsics

Fortran intrinsics like matmul and dot_product give relatively poor performance regardless of the level of compiler optimization.

-qessl: replace Fortran intrinsics with the equivalent routine from the ESSL library.

Must link with **–lessl** (single threaded) or **–lesslsmp** (multi-threaded).

For the multi-threaded version it uses the same number of threads as any ESSL or OpenMP routine in the code: 16 by default or the value of the environment variable **OMP_NUM_THREADS**.

Optimization Example: Matrix Multiply(1)

Multiply two 2500 by 2500 real*8 matrices.

Directly: **-O3 –qarch=pwr3 –qtune=pwr3 –qstrict**

Fortran: c(i,j)=c(i,j)+a(i,k)*b(k,j)

C: c[i][j]=a[i][k]*b[k][j]+c[i][j]

Performance depends on the order of the index variables.

	ijk	ikj	jik	jki	kij	kji
Fortran	18	6	15	171	6	154 MFlops
C	15	152	18	6	154	6 MFlops

Add **–qhot** to compile and performance differences disappear: Fortran: 446 MFlops, C: 410-413 MFlops for all indices.

Optimization Example: Matrix Multiply(2)

Add **–qsmp=auto** to compile and run dedicated with 16 threads.

- Fortran: 2368 MFlops.
- C: 2452 MFlops.

Fortran Intrinsic **matmul**:

- 171 MFlops.
- -qessl –lessl: 1234 MFlops.
- -qessl –lesslsmp: (16 threads) 18,323 MFlops.

ESSL routine DGEMM:

- -lessl: 1247 MFlops.
- -lesslsmp: (16 threads) 19,696 MFlops.

Other Useful Compiler Options

-Q+proc – Inline specific procedure proc.

-qmaxmem=n – Limits the amount of memory used by the compiler to n kilobytes. Default n=2048. n=-1 memory is unlimited.

-C or **–qcheck** – Check array bounds.

-g – Generate symbolic information for debuggers.

-v or **–V** – Verbosely trace the progress of compilations.

Fortran Livermore Loops

24 numeric kernels.

Written in “fairly straightforward, uncomplicated Fortran 77”.

8 different summary statistics returned including mean, minimum, and maximum MFlops for each kernel.

Fortran Livermore Loops Timings

Optimization Level	Compile Time	Mean	Min	Max MFlops
unoptimized	1.55	47	11	141
-O2	7.98	244	22	1071
-O3 –qarch=pwr3 –qtune=pwr3 -qstrict	13.52	290	52	1291
-O3 –qarch=pwr3 –qtune=pwr3	13.43	309	51	1292
-O3 –qarch=pwr3 –qtune=pwr3 -qhot	35.29	305	65	1216
-O4	40.41	314	**67**	1346
-O5	50.95	318	65	1312
-O4 -qnohot	26.58	**322**	55	**1475**
-O5 -qnohot	50.24	308	64	1214

NAS Kernels

Seven Fortran Kernels described at http://www.netlib.org/benchmark/nas-doc.

MXM – Matrix-matrix multiply.

CFFT2D – Complex 2D FFT.

CHOLSKY – Cholesky decomposition.

BTRIX – Tridiagonal solver.

GMTRY – Gaussian elimination.

EMIT – Creates new vortices according to certain boundary conditions.

VP – Invert 3 pentadiagonals simultaneously.

NAS Kernels

Optimization Level	Compile time (secs)	Average MFlops
unoptimized	.79	30.8
-O2	2.82	73.6
-O3 –qarch=pwr3 –qtune=pwr3 -qstrict	7.84	79.1
-O3 –qarch=pwr3 –qtune=pwr3	7.98	80.1
-O3 –qarch=pwr3 –qtune=pwr3 -qhot	26.56	95.8
-O4	28.08	94.7
-O5	28.35	**97.6**
-O4 -qnohot	13.58	79.5
-O5 -qnohot	26.81	93.4

Individual NAS Kernels MFlops

Opt	MXM	FFT	CHOL	BT	GM	EM	VP
None	53.3	34.8	18.3	56.5	12.6	95.9	25.3
-O2	235.8	162.7	136.2	210.4	17.5	153.1	36.9
-O3 -qstrict	**488.1**	161.4	189.7	213.0	17.9	151.7	39.1
-O3	488.0	160.8	203.5	214.6	17.8	156.2	40.3
-O3 -qhot	456.6	173.5	210.5	**309.8**	17.9	466.2	71.6
-O4	450.9	157.5	214.6	309.3	17.9	**469.7**	71.6
-O5	444.2	**175.0**	**223.5**	**309.8**	**18.4**	468.8	71.5
-O4 -qnohot	486.6	162.1	202.3	200.2	17.9	155.4	39.7
-O5 -qnohot	443.9	**175.0**	221.7	298.4	**18.4**	155.4	**71.9**

SMG2000 ASCI Purple Benchmark

Parallel semicoarsening multigrid solver.

SPMD code written in ISO-C using MPI.

Run on 8 processors of 1 node.

Timings are the “wall clock time” returned by the program.

SMG2000 Timings

Optimization Level	Compile Time	Elapsed Secs
unoptimized	38.4	91.7
-O2	69.2	28.3
-O3 –qarch=pwr3 –qtune=pwr3 -qstrict	87.3	28.1
-O3 –qarch=pwr3 –qtune=pwr3	83.0	27.9
-O3 –qarch=pwr3 –qtune=pwr3 –qhot	201.8	28.1
-O4	185.4	28.2
-O5	190.8	27.9
-O4 -qnohot	143.0	27.0
-O5 -qnohot	115.6	**26.8**

References

See the web page http://hpcf.nersc.gov/computers/SP/options.html for an expanded version of this presentation along with many references.

Finis

End of this presentation.